УДК 004

РАЗРАБОТКА СИСТЕМЫ ПОИСКА И ИНДЕКСИРОВАНИЯ КОНТЕНТА АУДИОЗАПИСЕЙ

Р. А. Климов[1] [0009-0003-8566-7946], А. Ш. Якупов[2] [0000-0002-2333-8819]

Казанский (Приволжский) федеральный университет, ул. Кремлевская, 35, г. Казань, 420008

[1]itis.klimov@gmail.com, [2]asyakupov@kpfu.ru

Аннотация

Статья посвящена разработке системы поиска и индексации аудиофайлов с использованием автоматического распознавания речи (ASR) и Elasticsearch. Проанализированы актуальные системы транскрибирования аудиофайлов на русском языке и выбрана система whisper как лучшая. Создан алгоритм оптимизации скорости транскрибирования с помощью параллелизации процессов обработки файла, продемонстрирована его эффективность. Построена система на микросервисной архитектуре, способная индексировать контент аудиофайлов и их метаданные для поиска. Результаты исследования показали, что предложенный подход может быть применен для создания эффективных и гибких систем поиска и аналитики аудиоинформации.

Ключевые слова*: транскрибирование, индексирование, параллелизация, микросервисы, масштабируемость.*

ВВЕДЕНИЕ

В современном мире, который характеризуется стремительным ростом информационных технологий и объемов данных, особое внимание уделяется аудиоинформации. В условиях постоянного увеличения количества аудиозаписей в образовании, медицине, музыкальной индустрии, журналистике и других областях актуальность и необходимость эффективного поиска по контенту аудиозаписей становятся все более очевидными. Это утверждение подтверждается различными исследованиями.

В конце 2021 года компания Amazon — один из крупнейших провайдеров хранилищ медиаконтента — создала и опубликовала в открытом доступе (open-

source) свою систему поиска по медиаконтенту mediaSearch [1]. Однако эта система специализируется на сервисах Amazon, которые в настоящее время невозможно использовать в России из-за санкций. В то же время, адаптация системы под open-source и такие аналоги, как openShift, является непростой задачей, так как архитектура системы не является модульной – все компоненты сильно связаны и не могут быть заменены аналогами без внесения изменений в исходные код и архитектуру.

В исследовании 2023 года [2] учеными из Бангладеша описана проблема отсутствия решений в сфере поиска по медиаконтенту. Они создают систему Sherlok, позволяющую выполнять поиск по содержимому документов и фотографий, индексируя данные одного из крупнейших open-source файлового хранилища — openShift Swift [3].

В исследовании 2022 года, опубликованном на ResearchGate [4], авторы акцентируют внимание на отсутствии практических решений в области поиска по контенту аудиозаписей и подчеркивают значимость дополнительных научных работ в этом направлении, в частности, в контексте библиотек и информационных наук (Library and Information Science, LIS).

Вышесказанное подчеркивает актуальность разработки системы поиска информации по контенту аудиозаписей, которая может заполнить существующий пробел в этой области и удовлетворить растущие потребности пользователей. Исследование и разработка системы, способной осуществлять поиск по содержанию слов в аудиофайлах, позволят пользователям быстро и точно находить нужную информацию, облегчая тем самым работу и улучшая качество принятия решений.

ВЫБОР КОМПОНЕНТОВ СИСТЕМЫ

Выбор архитектуры. При выборе архитектуры системы учитывались два ключевых требования, основанных на особенностях данной доменной области:

1) *Возможность гибкой замены компонентов*. Это важно, поскольку технологии транскрибирования аудио сегодня активно развиваются и улучшаются. Поэтому необходима архитектура, которая позволит быстро и без больших усилий

обновлять поисковую систему, заменяя отдельные ее компоненты без внесения изменений в исходный код остальных.

2) *Горизонтальное масштабирование*. Обработка аудио данных может включать работу с большими объемами информации, а транскрибирование требует значительных вычислительных ресурсов. Для обеспечения приемлемого времени обработки больших объемов данных необходимо обеспечить распределение вычислительных задач между различными сервисами или узлами.

Исходя из названных требований, мы выбрали микросервисную архитектуру. Её основные компоненты представлены на рисунке 1.

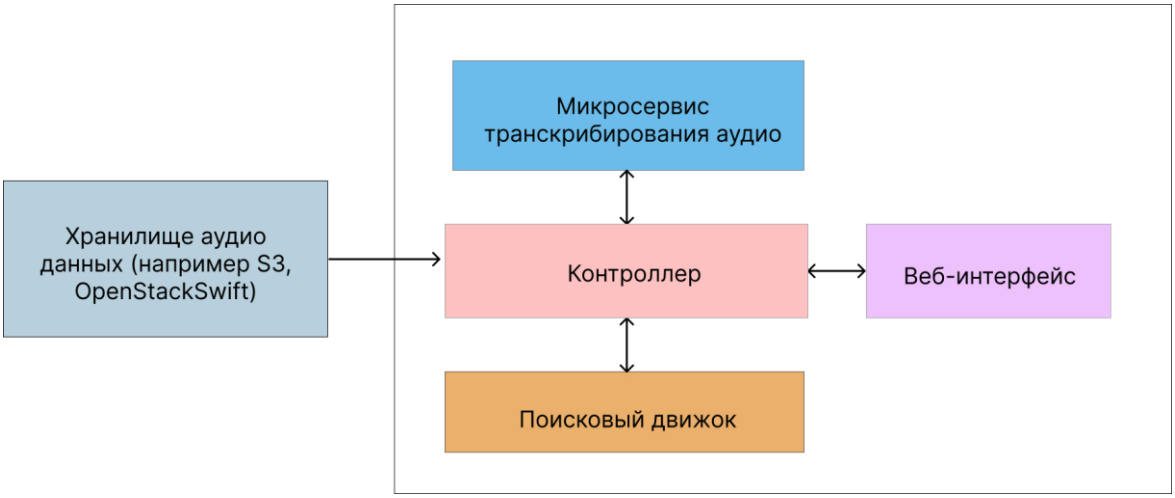

Рис. 1. Высокоуровневая архитектура системы.

Анализ решений для транскрибирования аудиофайлов. Для осуществления поиска релевантных файлов нужен инструмент конвертации аудио файлов в текст (транскрибирования). Поэтому следующим шагом исследования стал подбор оптимальной модели ASR (Automatic Speech Recognition). При таком подборе учитывались следующие критерии:

- Высокая точность распознавания – чем точнее перевод из аудио в текст, тем более релевантными будут результаты поиска;
- Поддержка русского языка – итоговый продукт ориентирован на рынок РФ, поэтому нужна поддержка русского языка, который для первой версии системы будет единственным;
- Доступность – наличие возможности использовать модель в РФ;

- Стоимость – у open-source решений будет преимущество перед платными аналогами.

Были рассмотрены 4 решения:

- Google Cloud Speech to Text [5];
- VOSK Speech Recognition Toolkit [6];
- Yandex SpeechKit [7];
- Whisper OpenAI [8].

Google Cloud Speech to Text поддерживает более чем 125 языков мира, в том числе русский, для конвертирования аудио в текст. Доступ осуществляется с использованием API.

Vosk Speech Recognition Toolkit – легковесная open source модель, которая поддерживает более 20 языков и диалектов и автономный режим работы, обеспечивая непрерывную транскрипцию большого словарного запаса.

Yandex SpeechKit – сервис распознавания и синтеза речи платформы Яндекс.Облако. Работает на базе разработанной в Яндекс технологии распознавания речи, поддерживает русский язык.

Whisper OpenAI – система применяется для транскрибирования и перевода, поддерживает около 10 языков, в том числе русский. Обучение модели Whisper проводилось на очень разнообразных данных, что позволяет распознавать не только чистую речь, но и с высокой долей точности понимать уникальные акценты, разговоры при фоновом шуме и технические термины.

В 2022 году опубликована статья [9], где проведено сравнение первых трех решений. Анализ точности транскрипции проведен на основе специализированных метрик WER (Word Error Rate), MER (Match Error Rate), WIL (Word Information Lost) [10]. В качестве экспериментальных данных были использованы две разноплановые аудиозаписи длительностью 20 секунд: запись университетской лекции и телефонный звонок рекламного агента. Для анализа и вычисления был использован язык программирования Python. Из результатов, приведенных в этой статье, видно, что Google Cloud Speech to Text демонстрирует лучшие показатели по всем трем метрикам. Поэтому вторым этапом сравнения был выбор между Google Cloud Speech to Text и Whisper OpenAI, который не был рассмотрен в [11].

Сравнение Google Cloud Speech to Text и Whisper OpenAI проводилось нами самостоятельно. За неимением доступа к данным, которые использовались в [9], для второго этапа сравнения были подобраны другие, но максимально схожие с первыми аудио файлы: разноплановые между собой, длительностью также 20 секунд и похожие по тематике. Два аудио файла представляли собой телефонный разговор с сотрудником банка и лекцию по психологии в университете. При подборе внимание уделялось тому, чтобы на записях присутствовали естественные шумы, а дикция говорящих была не идеальной.

Google Cloud Speech to Text позволяет транскрибировать аудио в текст онлайн через платформу Google Cloud. Для анализа Whisper OpenAI на языке программирования Python был написан скрипт, который использует методы системы Whisper для транскрибирования аудио.

Для анализа двух моделей были использованы те же метрики, что и в первом этапе анализа. Для вычисления метрик WER, MER, WIL была использована open-source библиотека JiWER [11]. Также к полученным текстам были применены методы препроцессинга из этой же библиотеки:

- приведение к нижнему регистру;
- удаление лишних пробелов и служебных символов (\t, \n, \r и т.д.);
- приведение текста к определенному виду для вычисления метрик.

В таблицах 1 и 2 приведены результаты вычисления метрик для двух аудиофайлов.

Таблица 1. Сравнение метрик для записи лекции по психологии

Модель	WER, %	MER, %	WIL, %
Google Cloud StT	72	72	86
Whisper OpenAI	33	33	51

Таблица 2. Сравнение метрик для телефонного разговора с сотрудником банка

Модель	WER, %	MER, %	WIL, %
Google Cloud StT	78	75	88
Whisper OpenAI	8	8	15

Из сравнения значений метрик видно, что Whisper OpenAI существенно более точен по сравнению с Google Cloud Speech to Text, из чего можно сделать предварительный вывод об оптимальности Whisper OpenAI для решения поставленной задачи. В завершение сравним две модели по оставшимся критериям.

Стоимость. Если оценивать затраты на использование, то Whisper OpenAI имеет преимущество, так как является открытым решением, в то время, как стоимость использования модели от Google стартует от $0.024 за минуту аудио файла.

Доступность. Даже если выбор будет сделан в пользу Google Cloud Speech to Text, на данный момент оплатить тариф не представляется возможным из-за санкций по отношению к РФ.

Вывод. По результатам сравнительного анализа четырех моделей Whisper OpenAI показал наименьшие значения ошибок в транскрипции по всем трем использованным метрикам. Кроме того, эта модель выигрывает и по остальным критериям, по которым проводилось сравнение, и поэтому является оптимальным вариантом ASR модели для использования на русском языке.

Оптимизация Whisper с использованием параллелизма. С момента выпуска официальной модели Whisper от OpenAI [8] появились несколько оптимизированных реализаций, включая whisper.cpp [12] и faster-whisper [13].

На данный момент самой эффективной реализацией является faster-whisper, эффективность которой достигается благодаря использованию CTranslate2 [14] – это библиотека на C++ и Python, предназначенная для эффективной работы с моделями Transformer в процессе вывода. Данный проект включает собственную среду выполнения, применяющую множество методов оптимизации производительности, таких как квантование весов, слияние слоев, переупорядочивание партий и т. д., что позволяет ускорить обработку и снизить потребление памяти моделей Transformer на CPU и GPU. Благодаря этому, faster-whisper работает до 4 раз быстрее официальной реализации, используя при этом меньше памяти [13].

Хотя текущая реализация faster-whisper обеспечивает возможность параллельного выполнения, она ограничена параллельным транскрибированием разных файлов, в то время как обработка одного файла происходит последовательно. Эта проблема может быть критичной в ряде ситуаций, когда требуется быстро получить транскрибацию одного конкретного файла, например, лекции.

Наше исследование направлено на устранение этого ограничения и повышение производительности транскрибирования путем разделения одного файла на части (чанки) и их параллельной обработки.

Стоит учесть, что в оригинальной модели Whisper используется способ подачи транскрибации предыдущего 30-секундного окна файла в качестве подсказки для следующего окна, чтобы получить непрерывную транскрибацию [15]. В случае разделения аудиофайла на чанки, это свойство не может быть использовано в оптимальной мере, так как контекст между частями теряется. Это может повлиять на качество и непрерывность транскрибации при обработке разделенных аудиофайлов.

Чтобы устранить этот недостаток, можно разделить аудиофайл на чанки, опираясь на результаты алгоритма определения активности голоса, который позволяет найти «независимые» фрагменты речи, разделенные 2–3 секундами тишины. Применение такого подхода позволит сохранить контекст и непрерывность

транскрибации между чанками, так как разделение будет происходить в моменты, когда речи нет, и связь между сегментами минимальна.

Описание алгоритма разделения файла на чанки:

1. С помощью python библиотеки ‘ffmpeg’ [16] к аудиофайлу применяется фильтр ‘silencedetect’ для определения участков тишины (настройки фильтра: уровень звука – 20 децибел, продолжительность тишины – 2 секунды). На выходе получаем массив, каждое значение в котором является временем начала «тишины» в аудиофайле;

2. Для каждого значения полученного массива вычисляется его «ценность» на основе его расстояния от начала и конца аудиофайла по формуле

$$W = point * (end\_point - point),$$

где *point* – время начала тишины, *end_point* – время конца аудиофайла.

Ценность точек, наиболее удаленных от начала и конца аудиофайла, обусловлена тем, что такое распределение позволяет создавать чанки примерно одинаковой длины. Это важно, поскольку в параллельной обработке оптимальное использование вычислительных мощностей достигается, когда все параллельные процессы имеют сбалансированную нагрузку. Если чанки имеют разную длину, короткие чанки будут обрабатываться быстрее, и вычислительные мощности, отведенные на их обработку, будут простаивать в ожидании завершения обработки длинных чанков другими процессами;

3. Значения массива сортируются в порядке убывания их ценности;

4. Выбираются *n*-1 первых значений массива, как «наиболее ценных», где *n* – количество доступных процессов для обработки;

5. Полученные значения сортируются в порядке возрастания их временных меток;

6. Аудиофайл разделяется на чанки с использованием выбранных точки тишины в качестве точек разделения.

Общая временная сложность алгоритма $O(m * log(m))$, где m – количество найденных точек «тишины».

Результаты тестирования оптимизации. Тесты выполнялись на MacBook с процессором Apple M1 Pro, 8 ядер, с 16 гигабайтами оперативной памяти с использованием модели “tiny” faster-whisper, версия сборки 0.4.1.

Для транскрибирования была выбрана аудиокнига длительностью 3706 секунд – 1 час, 1 минута и 46 секунд. Результаты представлены в таблице 3.

Таблица 3. Результаты тестирования оптимизации транскрибации

Ядра	Время	Скорость
1	134 сек. – 2 мин. 14 сек.	27.66x (без оптимизации)
2	86.56 сек. – 01 мин. 27 сек.	42.81x
3	59.6 сек.	62.18x
4	61 сек.	60.75x
5	59.5 сек.	62.29x
6	58 сек.	63.9x

Выводы. При переходе от одного ядра к двум производительность увеличивается значительно, с 27,66x до 42,81x (в 1.54 раза). Однако при дальнейшем увеличении количества ядер прирост производительности становится менее значительным, и в некоторых случаях даже наблюдается небольшое снижение производительности (например, при переходе от 3 к 4 ядрам). Это может быть вызвано неравномерным распределением участков «тишины» аудиофайла, что может приводить к простою вычислительных ресурсов.

Данный алгоритм можно улучшить, анализируя длительности чанков и подбирая для работы оптимальное количество вычислительных ядер.

Индексирование контента для поиска. В качестве поискового движка был выбран Elasticsearch – распределенный, бесплатный и открытый поисковый движок и система аналитики для всех типов данных, включая текстовые, числовые,

геопространственные, структурированные и неструктурированные. Высоко адаптивный API запросов Elasticsearch позволяет одновременно использовать фильтрацию, сортировку, постраничный вывод и агрегацию в одном запросе. Elasticsearch способен легко обрабатывать неструктурированные данные и индексировать файлы JSON. Он автоматически пытается определить сопоставления классов и корректирует новые или удаленные поля. Также он предлагает встроенную функциональность для кластеризации, репликации данных и мгновенного переключения при сбоях, которые прозрачны для пользователя [17].

В роли контроллера потока данных и точкой хода API для хранилищ данных выступает микросервис, написанный на java с использованием фреймворка Spring.

При первой попытке индексирования файла на контроллере происходит первичная настройка elasticsearch. В частности, создается индекс "transcriptions" с настройкой типа данных "completion" для поля текста. Такая конфигурация необходима для поддержки функциональности автодополнения и быстрого предложения результатов поиска на основе символов, введенных пользователем.

Взаимодействие между микросервисами осуществляется с помощью протокола gRPC.

Описание потока индексирования аудиофайла:

1. Файл вместе с метаинформацией передается в систему через эндпоинт /upload по REST или gRPC;

2. Генерируется внутренний идентификатор файла и добавляется в очередь вместе с метаинформацией файла (например, ссылкой на файл);

3. Файл вместе с идентификатором асинхронно передается на микросервис транскрибирования по gRPC;

4. После получения ответа из микросервиса транскрибирования транскрипция вместе с информацией помещается для индексирования в elasticsearch.

РЕЗУЛЬТАТЫ

В результате соединения выбранных подходящих компонентов была разработана гибкая и масштабируемая система, способная:

- Эффективно индексировать контент аудиофайлов и их метаданные;
- Выполнять поиск по контенту аудиофайлов;
- Находить для выбранной аудиозаписи наиболее «близкие» по контенту с использованием параметра “more_like_this” [18] elasticsearch.

Для демонстрации работы API системы был разработан простой веб-интерфейс, представленный на рис. 2.

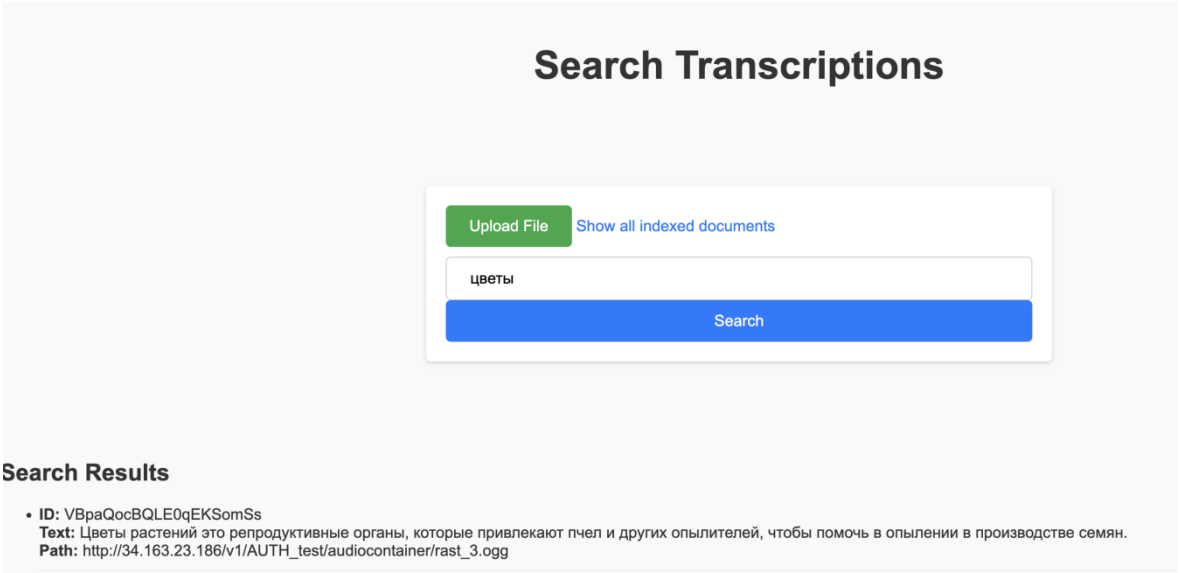

Рис. 2. Веб-интерфейс системы

ЗАКЛЮЧЕНИЕ

Проведен анализ существующих решений для транскрибации аудио на русском языке, в результате чего было выбрано наиболее эффективное и актуальное решение. Далее был разработан алгоритм для оптимизации скорости транскрибации, основанный на параллелизации процессов, и продемонстрирована его эффективность.

Как результат, создана масштабируемая микросервисная архитектура, позволяющая эффективно индексировать контент аудиозаписей для поиска.

В дальнейшей перспективе планируется продолжить оптимизацию скорости транскрибации, а также сосредоточиться на улучшении пользовательского

опыта за счет добавления инструментов быстрого анализа полученных данных в процессе поиска.

СПИСОК ЛИТЕРАТУРЫ

1. AWS Kendra Transcribe Media Search.
URL: https://github.com/aws-samples/aws-kendra-transcribe-media-search

2. *Noor J., Rownak A., Ratul R., Mondal J.* Sherlok in OSS: A Novel Approach of Content-Based Searching on Object Storage System. 2023.
URL: https://arxiv.org/pdf/2303.02105.pdf.

3. Swift Object Storage.
URL: https://www.openstack.org/software/releases/zed/components/swift

4. *Adrakatti A., Mulia K.R.* Research Challenges of Library and Information Science in retrieving content based Multimedia Information. 2023.
URL:https://www.researchgate.net/publication/361107734_Research_Challenges_of_Library_and_Information_Science_in_retrieving_content_based_Multimedia_Information.

5. Google Speech. URL: https://console.cloud.google.com/speech/overview.

6. Vosk. URL: https://github.com/alphacep/vosk.

7. Yandex SpeechKit. URL: https://cloud.yandex.com/en/services/speechkit.

8. Whisper. URL: https://github.com/openai/whisper.

9. *Подопригорова Н. С., Подопригорова С. С., Кан А. Д.* Автоматическое распознавание речи в системе информационного поиска по аудио // Искусственный интеллект в автоматизированных системах управления и обработки данных, Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет). 2022. Т. 2. С. 339–345.

10. *Morris A., Maier V., Green P.* From WER and RIL to MER and WIL. 2004.
URL:https://www.isca-speech.org/archive_v0/archive_papers/interspeech_2004/i04_2765.pdf.

11. JiWER: A Simple and Fast Python Package to Evaluate an Automatic Speech Recognition System. URL: https://github.com/jitsi/jiwer

12. Whisper.cpp. URL: https://github.com/ggerganov/whisper.cpp

13. Faster-whisper. URL: https://github.com/guillaumekln/faster-whisper

14. CTranslate2. URL: https://github.com/OpenNMT/CTranslate2/
15. Prompt vs prefix in DecodingOptions.
URL: https://github.com/openai/whisper/discussions/117
16. FFmpeg. URL: https://ffmpeg.org/
17. ElasticSearch. URL: https://www.elastic.co/
18. ElasticSearch More like this query
URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html

DEVELOPMENT OF A SYSTEM FOR SEARCHING AND INDEXING THE CONTENT OF AUDIO RECORDINGS

R. A Klimov[1] [0009-0003-8566-7946], A. Sh. Yakupov[2] [0000-0002-2333-8819]

[1, 2] *Kazan (Volga Region) Federal University, 35 Kremlevskaya str., Kazan, 420008*

[1]itis.klimov@gmail.com, [2]asyakupov@kpfu.ru

Abstract

The article is devoted to the development of a search and indexing system for audio files using Automatic Speech Recognition (ASR) and Elasticsearch. Current Russian-language audio file transcription systems have been analyzed, and Whisper has been chosen as the best one. An algorithm for optimizing transcription speed using parallelization of file processing processes has been developed, and its effectiveness has been demonstrated. A microservice architecture-based system has been built, capable of indexing audio file content and their metadata for search purposes. The research results show that the proposed approach can be applied to create efficient and flexible systems for searching and analyzing audio information.

Keywords: *transcription, indexing, parallelization, microservices, scalability.*

REFERENCES

1. AWS Kendra Transcribe Media Search.
URL: https://github.com/aws-samples/aws-kendra-transcribe-media-search

2. *Noor J., Rownak A., Ratul R., Mondal J.* Sherlok in OSS: A Novel Approach of Content-Based Searching on Object Storage System. 2023.
URL: https://arxiv.org/pdf/2303.02105.pdf.

3. Swift Object Storage.
URL: https://www.openstack.org/software/releases/zed/components/swift

4. *Adrakatti A., Mulia K. R.* Research Challenges of Library and Information Science in retrieving content based Multimedia Information.
URL:https://www.researchgate.net/publication/361107734_Research_Challenges_of_Library_and_Information_Science_in_retrieving_content_based_Multimedia_Information.

5. Google Speech. URL: https://console.cloud.google.com/speech/overview.

6. Vosk. URL: https://github.com/alphacep/vosk.

7. Yandex SpeechKit. URL: https://cloud.yandex.com/en/services/speechkit.

8. Whisper. URL: https://github.com/openai/whisper.

9. *Podoprigorova N. S., Podoprigorova S. S., Kan A. D.* Automatic Speech Recognition in an Information Retrieval System for Audio // Artificial Intelligence in Automated Control Systems and Data Processing, Bauman Moscow State Technical University (National Research University), 2022. V. 2. P. 339–345.

10. *Morris A., Maier V., Green P.* From WER and RIL to MER and WIL. 2004.
URL:https://www.isca-speech.org/archive_v0/archive_papers/interspeech_2004/i04_2765.pdf.

11. JiWER: A Simple and Fast Python Package to Evaluate an Automatic Speech Recognition System. URL: https://github.com/jitsi/jiwer

12. Whisper. URL: https://github.com/openai/whisper

13. Whisper.cpp. URL: https://github.com/ggerganov/whisper.cpp

14. Faster-whisper. URL: https://github.com/guillaumekln/faster-whisper

15. CTranslate2. URL: https://github.com/OpenNMT/CTranslate2/

16. Prompt vs prefix in DecodingOptions.
URL: https://github.com/openai/whisper/discussions/117

17. FFmpeg. URL: https://ffmpeg.org/

18. ElasticSearch. URL: https://www.elastic.co/

19. ElasticSearch More like this query

URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html

СВЕДЕНИЯ ОБ АВТОРАХ

КЛИМОВ Роман Алексеевич – магистрант, Казанский (Приволжский) федеральный университет, г. Казань.

Roman Aleckseevich KLIMOV is Master's student, Kazan (Volga region) Federal University,

email: itis.klimov@gmail.com;
ORCID: 0009-0003-8566-7946

ЯКУПОВ Азат Шавкатович – старший преподаватель, Казанский (Приволжский) федеральный университет, г. Казань.

Azat Shavkatovich YAKUPOV – Senior Lecturer, Kazan (Volga region) Federal University, Kazan.

email: asyakupov@kpfu.ru
ORCID: 0000-0002-2333-8819

Материал поступил в редакцию 17 апреля 2023 года